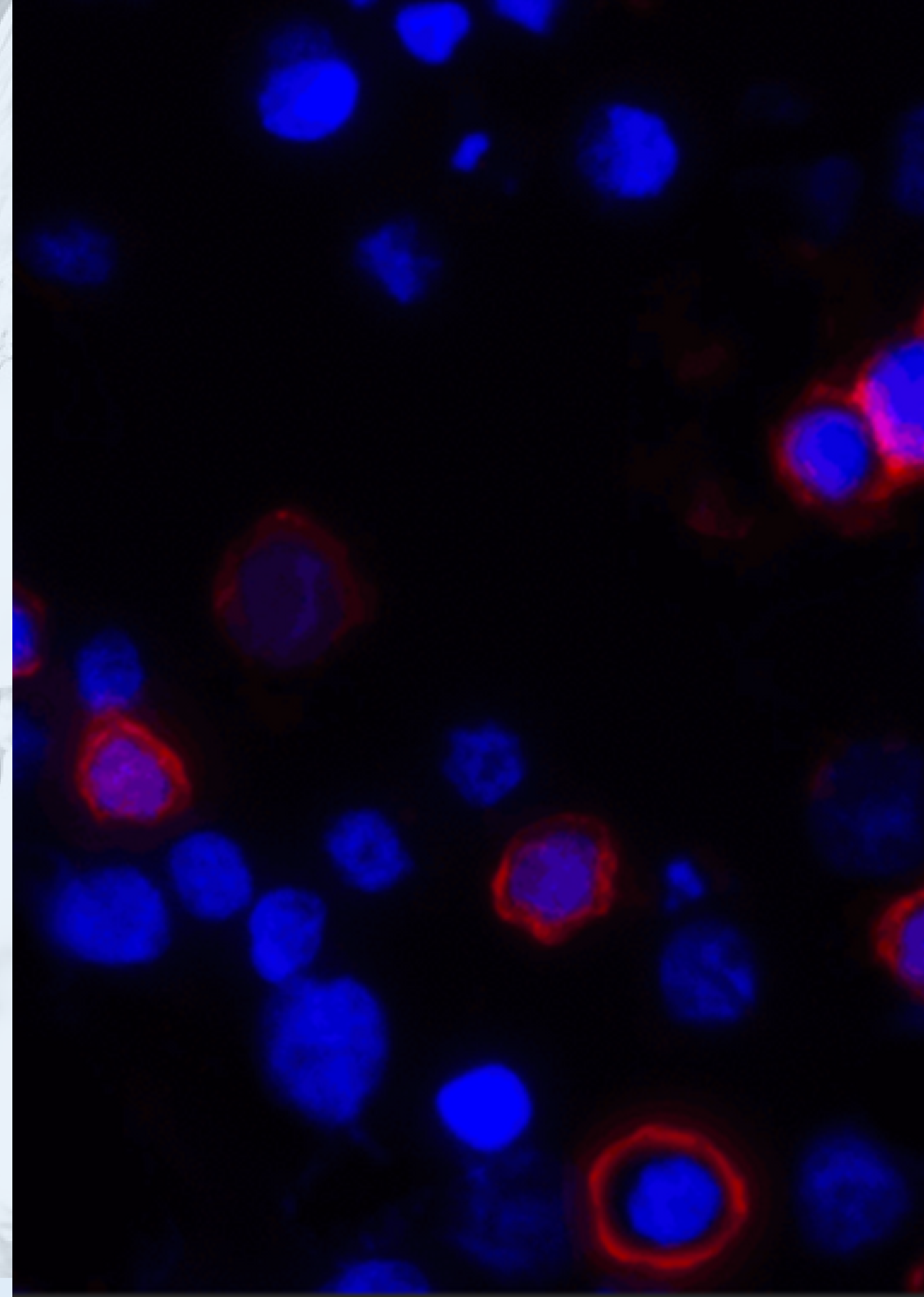
Image segmentation project

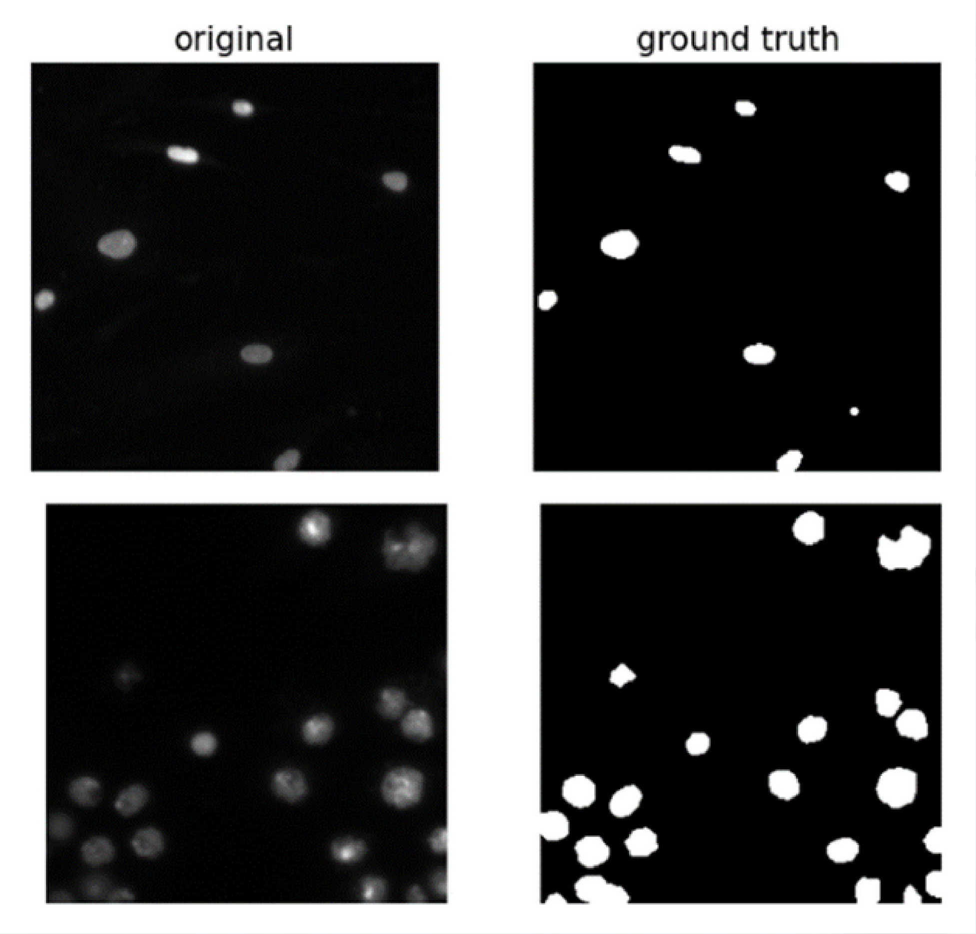
presented by

- Rivka Mcgowan
- Tzofiya Barel

project guide by

Dr. Moshe Amitay

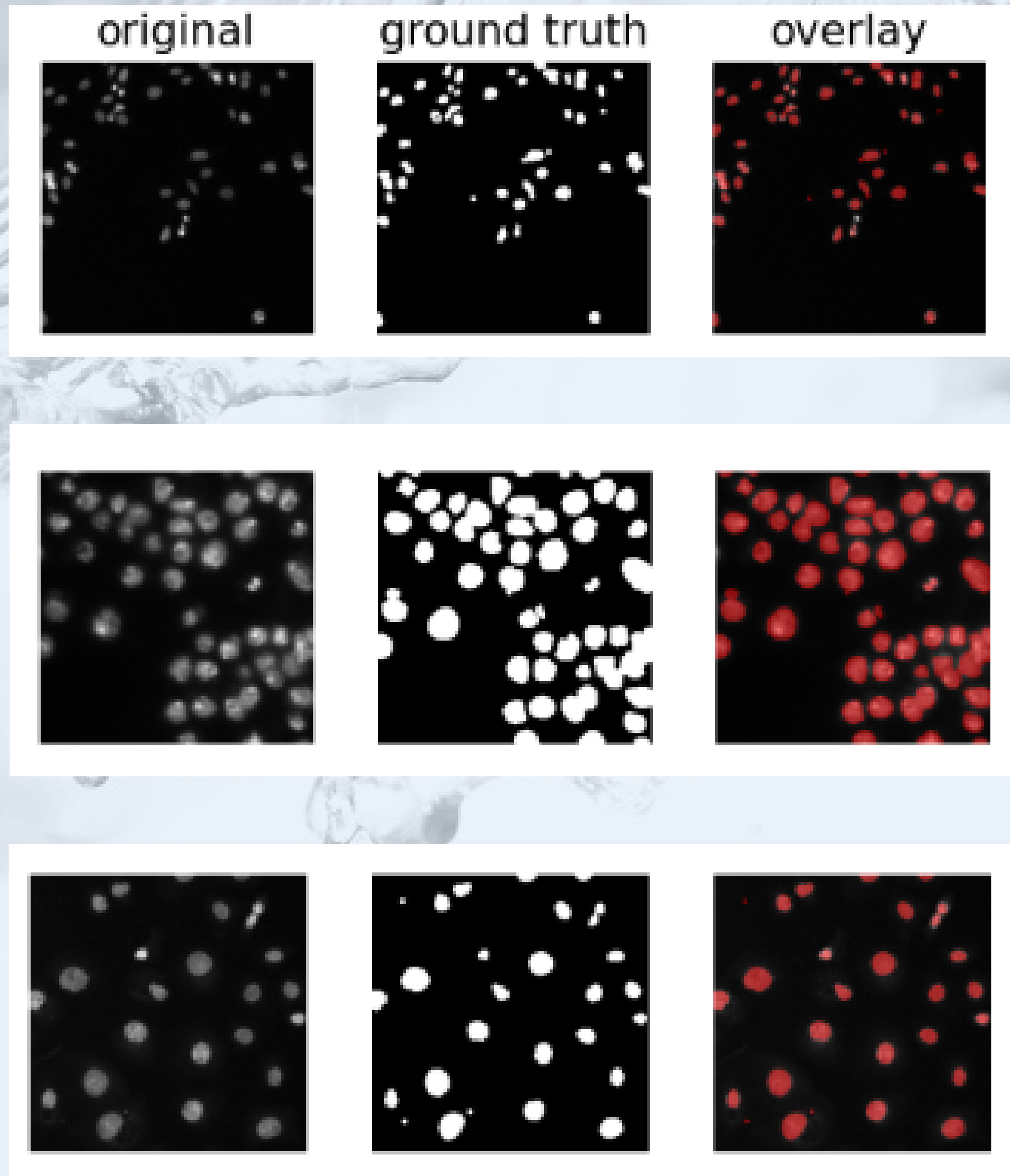


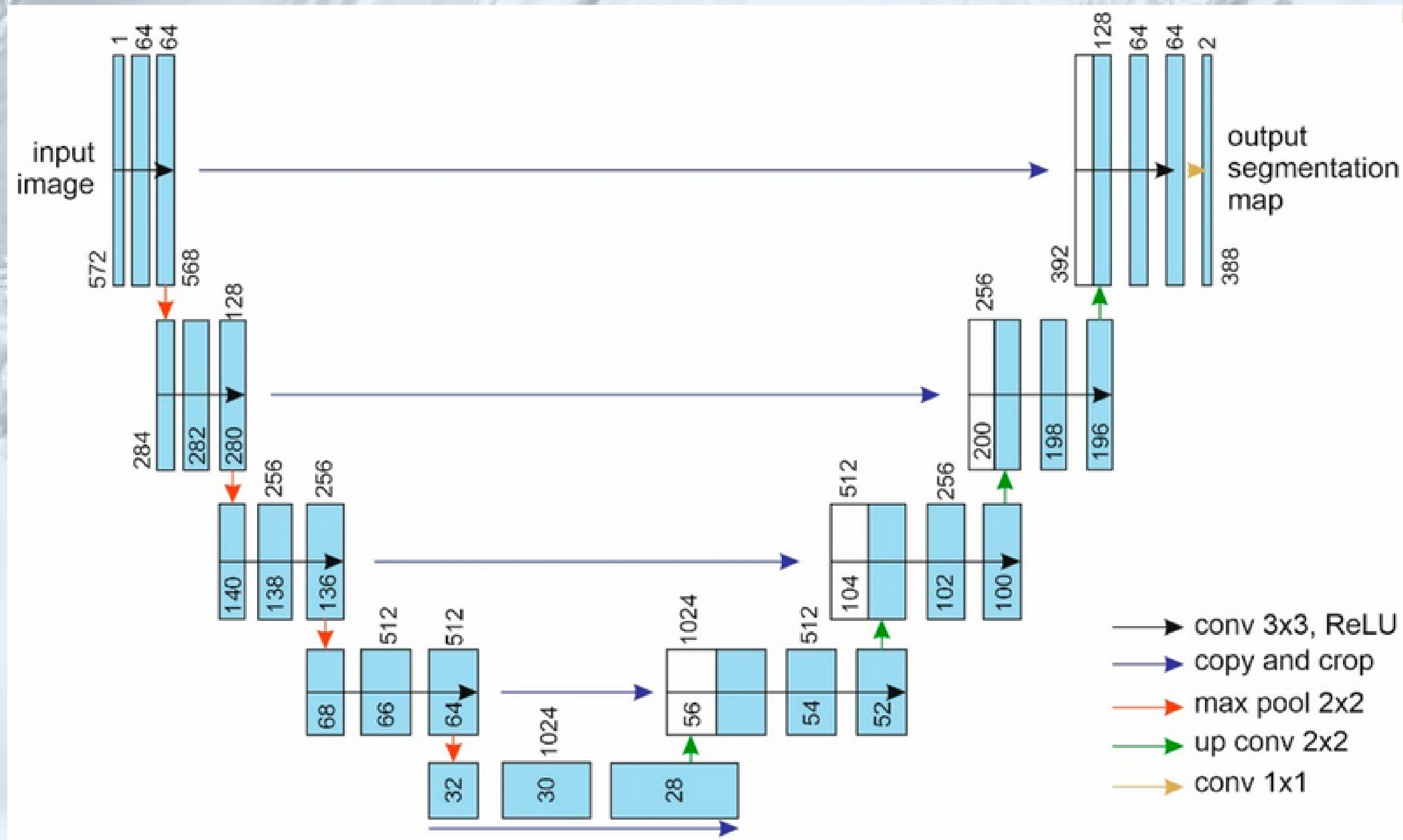


project target

The project will train a model that will know how to mark cells in an unclear image and create a suitable mask for them using a neural network.

Pre-processing of the Data


original
ground truth
overlay



מודל U-Net מחלק תמונה לחלקים שונים ומסווג כל פיקסל לאיזה חלק הוא שייך.
המודל משתמש בסוג מיוחד של בינה מלאכותית הנקראת רשת עצבית
קונבולוציונית. יש לו שני חלקים עיקריים. חלק שבו המודל מסתכל על התמונה
ומוצא תכונות חשובות, כמו קצוות, צורות ומרקמים. ואז דוחס את התמונה תוך
שמירה על המידע החשוב. ובחלק השני הוא לוקח את המידע הדחוס ומנסה
לשחזר את התמונה המקורית על ידי הוספת פרטים בחזרה.

U-Net model

Name of model: get_unet_128
Number of layers in the neural-net: 9 layers

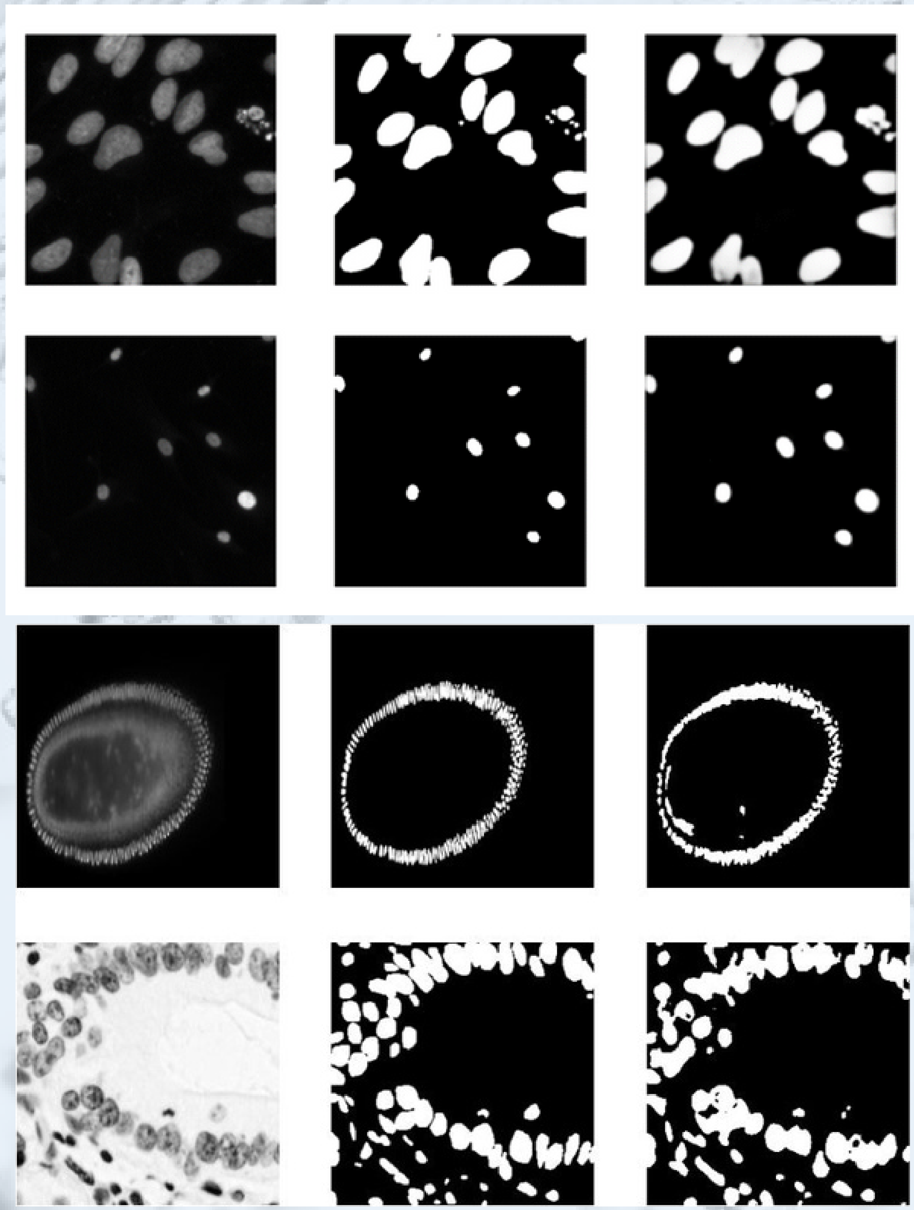
Distribution of the data
train: 70%
validation: 70% of the 30%
test: 30% of the 30%

Parameters of the model:
optimizer=SGD(lr=0.01, momentum=0.99)
loss='binary_crossentropy',
metrics=[iou, iou_thresholded]

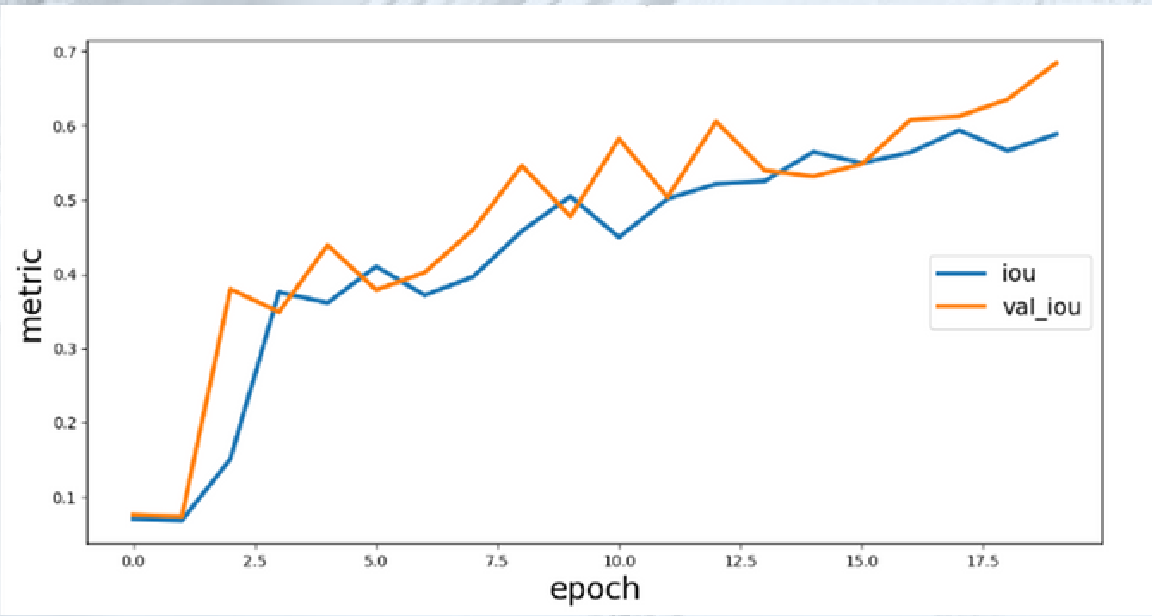
Average IoU: 0.7651381108179167
loss: 0.094

```
conv1 = Conv2D(32, 3, activation='relu', padding='same')(conv1)
pool1 = MaxPooling2D(pool_size=(2, 2))(conv1)

conv2 = Conv2D(64, 3, activation='relu', padding='same')(pool1)
conv2 = Conv2D(64, 3, activation='relu', padding='same')(conv2)
pool2 = MaxPooling2D(pool_size=(2, 2))(conv2)

conv3 = Conv2D(128, 3, activation='relu', padding='same')(pool2)
conv3 = Conv2D(128, 3, activation='relu', padding='same')(conv3)
pool3 = MaxPooling2D(pool_size=(2, 2))(conv3)

conv4 = Conv2D(256, 3, activation='relu', padding='same')(pool3)
conv4 = Conv2D(256, 3, activation='relu', padding='same')(conv4)
pool4 = MaxPooling2D(pool_size=(2, 2))(conv4)

conv5 = Conv2D(512, 3, activation='relu', padding='same')(pool4)
conv5 = Conv2D(512, 3, activation='relu', padding='same')(conv5)

# expansive path
up6 = Conv2DTranspose(256, 2, strides=(2, 2), padding='same')(conv5
up6 = concatenate([up6, conv4])
conv6 = Conv2D(256, 3, activation='relu', padding='same')(up6)
conv6 = Conv2D(256, 3, activation='relu', padding='same')(conv6)

up7 = Conv2DTranspose(128, 2, strides=(2, 2), padding='same')(conv6
up7 = concatenate([up7, conv3])
conv7 = Conv2D(128, 3, activation='relu', padding='same')(up7)
conv7 = Conv2D(128, 3, activation='relu', padding='same')(conv7)

up8 = Conv2DTranspose(64, 2, strides=(2, 2), padding='same')(conv7)
up8 = concatenate([up8, conv2])
conv8 = Conv2D(64, 3, activation='relu', padding='same')(up8)
conv8 = Conv2D(64, 3, activation='relu', padding='same')(conv8)

up9 = Conv2DTranspose(32, 2, strides=(2, 2), padding='same')(conv8)
up9 = concatenate([up9, conv1])
conv9 = Conv2D(32, 3, activation='relu', padding='same')(up9)
```

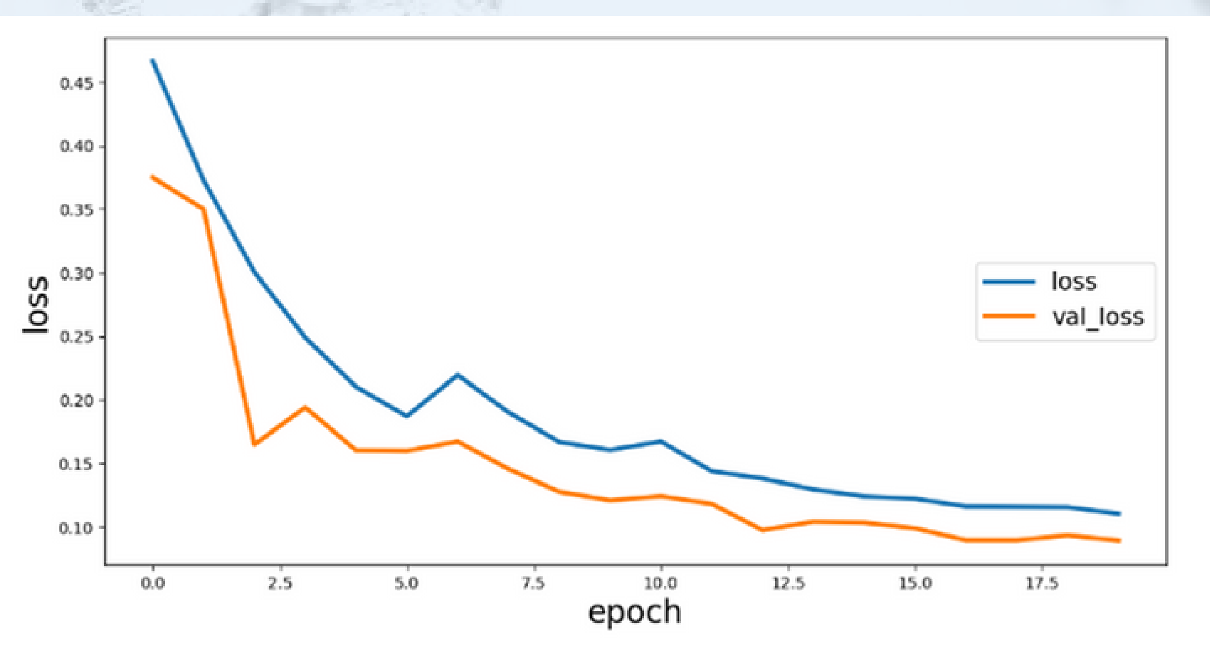
Results of the model



results



0.7
0.6
0.5
0.4
0.3
0.2
0.1
metric
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
epoch
iou
val_iou



0.45
0.40
0.35
0.30
0.25
0.20
0.15
0.10
loss
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
epoch
loss
val_loss

SUMMARY

Throughout the project we tried to run several models on our data and try to find the model that creates the most accurate match between the images and the masks.
We saw that the get_unet_128 model reaches the best performance we found - its loss is the lowest and you can see that it brings the best results when looking at the predicted mask in relation to the original mask.

Thanks for listening